

Python: module vcs.Canvas

vcs.Canvas

[index](#)

Canvas: the class representing a vcs drawing window
Normally, created by `vcs.init()`
Contains the method `plot`.

Modules

MA	vcs.boxfill	vcs.line	tempfile
MV	cdms	vcs.marker	vcs.template
Numeric	cdtime	vcs.meshfill	vcs.textcombined
RandomArray	cdutil	os	vcs.textorientation
Tkinter	vcs.colormap	vcs.outfill	vcs.texttable
vcs.animationgui	vcs.continents	vcs.outline	thread
vcs.colormapgui	copy	vcs.projection	types
vcs.graphicsmethodgui	vcs.displayplot	vcs.scatter	vcs
vcs.gui template editor	vcs.fillarea	signal	vcs.vector
vcs.pagegui	genutil	string	vcs.xvsvy
vcs.projectiongui	vcs.isofill	sys	vcs.xyvsvy
vcs. vcs	vcs.isoline	vcs.taylor	vcs.yxvsvx

Classes

[Canvas](#)
[animate_obj](#)

class *Canvas*

```
Function: Canvas # Construct a VCS Canvas class
```

Description of Function:

```
Construct the VCS Canas object. There can only be at most 8 VCS  
Canvases open at any given time.
```

Example of Use:

```
a=vcs.Canvas() # This examples constructs a V
```

Methods defined here:

```
BLOCK_X_SERVER(self, *args)
```

```
#####  
#
```

```

# Block the X server. It may NOT process do X11 commands.
#
#####

SCREEN_MODE(self, *args)
#####
#
# Return the Screen mode, either data mode or template editor
#
#####

UNBLOCK_X_SERVER(self, *args)
#####
#
# Unblock the X server. It may now proceed to do X11 commands
#
#####

__init__(self, mode=1, pause_time=0, call_from_gui=0)
#####
#
# Initialize the VCS Canvas and set the Canvas mode to 0. Bec
# is set to 0, the user will have to manually update the VCS
# using the "update" function.
#
#####

__setattr__(self, name, value)
#####
#
# Set attributes for VCS Canvas Class (i.e., set VCS Canvas M
#
#####

backing_store(self, *args)
Function: backing_store

Description of Function:
    This function creates a backing store pixmap for the VCS C

Example of Use:
    a=vcs.init()
    a.backing_store()

boxfill(self, *args, **parms)
Function: boxfill # Generate a boxfill

Description of Function:
    Generate a boxfill plot given the data, boxfill graphics m
    template. If no boxfill class object is given, then the 'd
    graphics method is used. Similarly, if no template class o
    then the 'default' template is used.

```

Example of Use:

```
a=vcs.init()
a.show('boxfill') # Show all the ex
box=a.getboxfill('quick') # Create instance
a.boxfill(array,box) # Plot array using spe
# template
templt=a.gettemplate('AMIP') # Create an instance of
a.clear() # Clear VCS canvas
a.boxfill(array,box,template) # Plot array using spe
a.boxfill(box,array,template) # Plot array using spe
a.boxfill(template,array,box) # Plot array using spe
a.boxfill(template,array,box) # Plot array using spe
a.boxfill(array,'AMIP','quick') # Use 'AMIP' template
a.boxfill('AMIP',array,'quick') # Use 'AMIP' template
a.boxfill('AMIP','quick',array) # Use 'AMIP' template
```

canvasid(self, *args)

Function: canvasid

Description of Function:

Return VCS Canvas object ID. This ID number is found at the end of the window title as part of its title.

Example of Use:

```
a=vcs.init()
a.open()
id = a.canvasid()
```

canvasinfo(self, *args)

Function: canvasinfo

Description of Function:

Obtain the current attributes of the VCS Canvas window.

Example of Use:

```
a=vcs.init()
a.plot(array,'default','isofill','quick')
a.canvasinfo()
```

canvasraised(self, *args)

Function: canvasraised

Raise the VC

Description of Function:

This function marks a VCS Canvas as eligible to be displayed. It moves the window to the top of the stack of its siblings.

Example of Use:

```
a=vcs.init()
...
a.canvasraised()
```

cgm(self, *args)

Function: cgm

Description of Function:

To save a graphics plot in CDAT the user can call CGM along with the output. This routine will save the displayed image on a binary vector graphics that can be imported into MSWord files are in ISO standards output format.

The CGM command is used to create or append to a cgm file. for saving a cgm file: 'Append' mode (a) appends cgm output file; 'Replace' (r) mode overwrites an existing cgm file w The default mode is to overwrite an existing cgm file (i.e

Example of Use:

```
a=vcs.init()
a.plot(array, 'default', 'isofill', 'quick')
a.cgm(o)
a.cgm('example') # by default a cgm file will ov
a.cgm('example', 'a') # 'a' will instruct cgm to append to
a.cgm('example', 'r') # 'r' will instruct cgm to overwrite
```

clear(self, *args)

Function: clear

Description of Function:

In VCS it is necessary to clear all the plots from a page. will clear all the VCS displays on a page (i.e., the VCS C

Example of Use:

```
a=vcs.init()
a.plot(array, 'default', 'isofill', 'quick')
a.clear()
```

close(self, *args)

Function: close

Description of Function:

Close the VCS Canvas. It will not deallocate the VCS Canvas. To deallocate the VCS Canvas, use the destroy method.

Example of Use:

```
a=vcs.init()
a.plot(array, 'default', 'isofill', 'quick')
a.close()
```

colormapgui(self, gui_parent=None, transient=0, max_intensity=None)

Function: colormapgui

Description of Function:

Run the VCS colormap interface.

The `colormapgui` command is used to bring up the VCS colormap GUI. This command is used to select, create, change, or remove colormaps.

Example of Use:

```
a=vcs.init()
a.colormapgui()
a.colormapgui(max_intensity = 255)
```

continents(self, *args, **parms)
Function: `continents` # Generate a continents plot

Description of Function:

Generate a continents plot given the data, continents graphics method, and template. If no continents class object is given, then the `continents` graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('continents') # Show all the existing continents
# methods
con=a.getcontinents('quick') # Create instance of continents class
a.continents(array, con) # Plot array using specified
# template
a.clear() # Clear VCS canvas
a.continents(array, con, template) # Plot array using specified
# template
```

createboxfill(self, Gfb_name=None, Gfb_name_src='default')
Function: `createboxfill` # Construct a new boxfill graphics method

Description of Function:

Create a new boxfill graphics method given the the name and the boxfill graphics method to copy the attributes from. If no boxfill graphics method name is given, then the default boxfill graphics method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then an error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('boxfill')
box=a.createboxfill('example1',)
a.show('boxfill')
box=a.createboxfill('example2', 'quick')
a.show('boxfill')
```

createcolormap(self, Cp_name=None, Cp_name_src='default')
Function: `createcolormap` # Construct a new colormap

Description of Function:

Create a new colormap secondary method given the the name colormap secondary method to copy the attributes from. If secondary method name is given, then the default colormap will be used as the secondary method to which the attributes are copied from.

If the name provided already exists, then a error will be raised. Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
cp=a.createcolormap('example1',)
a.show('colormap')
cp=a.createcolormap('example2','AMIP')
a.show('colormap')
```

createcontinents(self, Gcon_name=None, Gcon_name_src='default')

Function: createcontinents # Construct a new continents graphics method

Description of Function:

Create a new continents graphics method given the the name continents graphics method to copy the attributes from. If continents graphics method name is given, then the default method will be used as the graphics method to which the attributes are copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('continents')
con=a.createcontinents('example1',)
a.show('continents')
con=a.createcontinents('example2','quick')
a.show('continents')
```

createfillarea(self, name=None, name_src='default', style=None, index=None, color=None, priority=None)

Function: createfillarea # Construct a new fillarea secondary method

Description of Function:

Create a new fillarea secondary method given the the name fillarea secondary method to copy the attributes from. If secondary method name is given, then the default fillarea method will be used as the secondary method to which the attributes are copied from.

If the name provided already exists, then a error will be raised. Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
```

```

a.show('fillarea')
fa=a.createfillarea('example1',)
a.show('fillarea')
fa=a.createfillarea('example2','black')
a.show('fillarea')
fa2=a.createmarker(name='new', name_src='red',style=1, ind
                    color=242, priority=1, viewport=[0, 2.0, 0,
                    worldcoordinate=[0,100, 0,50]
                    x=[0,20,40,60,80,100],
                    y=[0,10,20,30,40,50] )      # Create instance
a.fillarea(fa2)                                # Plot using specificie

```

createisofill(self, Gfi_name=None, Gfi_name_src='default')

Function: createisofill # Construct a new isofill graphics method

Description of Function:

Create a new isofill graphics method given the the name and the name of the isofill graphics method to copy the attributes from. If no name of the isofill graphics method name is given, then the default isofill graphics method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. The method names must be unique.

Example of Use:

```

a=vcs.init()
a.show('isofill')
iso=a.createisofill('example1',)
a.show('isofill')
iso=a.createisofill('example2','quick')
a.show('isofill')

```

createisoline(self, Gi_name=None, Gi_name_src='default')

Function: createisoline # Construct a new isoline graphics method

Description of Function:

Create a new isoline graphics method given the the name and the name of the isoline graphics method to copy the attributes from. If no name of the isoline graphics method name is given, then the default isoline graphics method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. The method names must be unique.

Example of Use:

```

a=vcs.init()
a.show('isoline')
iso=a.createisoline('example1',)
a.show('isoline')

```

```
iso=a.createisoline('example2','quick')
a.show('isoline')
```

createline(self, name=None, name_src='default', ltype=None, width=None, color=None, priority=1)
Function: createline # Construct a new

Description of Function:

Create a new line secondary method given the the name and line secondary method to copy the attributes from. If no line secondary method name is given, then the default line secondary method will be used as the secondary method to which the attributes are copied from.

If the name provided already exists, then a error will be raised. Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('line')
ln=a.createline('example1',)
a.show('line')
ln=a.createline('example2','black')
a.show('line')
ln2=a.createline(name='new', name_src='red',ltype='dash',
                 color=242, priority=1, viewport=[0, 2.0, 0,
                 worldcoordinate=[0,100, 0,50]
                 x=[0,20,40,60,80,100],
                 y=[0,10,20,30,40,50] ) # Create instance
a.line(ln2) # Plot using specified line
```

createmarker(self, name=None, name_src='default', mtype=None, size=None, color=None, priority=1)
Function: createmarker # Construct a new

Description of Function:

Create a new marker secondary method given the the name and marker secondary method to copy the attributes from. If no marker secondary method name is given, then the default marker secondary method will be used as the secondary method to which the attributes are copied from.

If the name provided already exists, then a error will be raised. Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('marker')
mrk=a.createmarker('example1',)
a.show('marker')
mrk=a.createmarker('example2','black')
a.show('boxfill')
mrk2=a.createmarker(name='new', name_src='red',mtype='dash',
                    color=242, priority=1, viewport=[0, 2.0, 0,
```

```
worldcoordinate=[0,100, 0,50]
x=[0,20,40,60,80,100],
y=[0,10,20,30,40,50] )      # Create instance
a.marker(mrk2)                # Plot using specified
```

createmeshfill(self, Gfm_name=None, Gfm_name_src='default')

Function: createmeshfill # Construct a new meshfill

Description of Function:

Create a new meshfill graphics method given the the name and a meshfill graphics method to copy the attributes from. If no meshfill graphics method name is given, then the default meshfill method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('meshfill')
mesh=a.createmeshfill('example1',)
a.show('meshfill')
mesh=a.createmeshfill('example2','quick')
a.show('meshfill')
```

createoutfill(self, Gfo_name=None, Gfo_name_src='default')

Function: createoutfill # Construct a new outfill

Description of Function:

Create a new outfill graphics method given the the name and an outfill graphics method to copy the attributes from. If no outfill graphics method name is given, then the default outfill method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('outfill')
out=a.createoutfill('example1',)
a.show('outfill')
out=a.createoutfill('example2','quick')
a.show('outfill')
```

createoutline(self, Go_name=None, Go_name_src='default')

Function: createoutline # Construct a new outline

Description of Function:

Create a new outline graphics method given the the name and the outline graphics method to copy the attributes from. If no outline graphics method name is given, then the default outline graphics method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('outline')
out=a.createoutline('example1',)
a.show('outline')
out=a.createoutline('example2','quick')
a.show('outline')
```

createprojection(self, Proj_name=None, Proj_name_src='default')

Function: createprojection # Construct a new projection

Description of Function:

Create a new projection method given the the name and the projection method to copy the attributes from. If no exist projection method name is given, then the default projection method will be used as the projection method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('projection')
p=a.createprojection('example1',)
a.show('projection')
box=a.createprojection('example2','quick')
a.show('projection')
```

createscatter(self, GSp_name=None, GSp_name_src='default')

Function: createscatter # Construct a new scatter

Description of Function:

Create a new scatter graphics method given the the name and the scatter graphics method to copy the attributes from. If no scatter graphics method name is given, then the default scatter graphics method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('scatter')
sct=a.createscatter('example1',)
a.show('scatter')
sct=a.createscatter('example2','quick')
a.show('scatter')
```

createtaylordiagram(self, Gtd_name=None, Gtd_name_src='default')

Function: *createtaylordiagram* # Construct a new taylor diagram

Description of Function:

Create a new taylor diagram graphics method given the the name of the existing taylor diagram graphics method to copy the attributes from. If the name of the taylor diagram graphics method name is given, then the default method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('taylor diagram')
td=a.createtaylordiagram('example1',)
a.show('taylor diagram')
td=a.createtaylordiagram('example2','quick')
a.show('taylor diagram')
```

createtemplate(self, P_name=None, P_name_src='default')

Function: *createtemplate* # Construct a new template

Description of Function:

Create a new template given the the name and the existing template name. If the name of the existing template name is given, then the default template will be used as the template to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Template names must be unique.

Example of Use:

```
a=vcs.init()
a.show('template') # Show all the existing templates
con=a.createtemplate('example1') # create 'example1' template
a.show('template') # Show all the existing templates
con=a.createtemplate('example2','quick') # create 'example2' template
a.listelements('template') # Show all the elements of the template
```

createtext = *createtextcombined*(self, Tt_name=None, Tt_name_src='default', To_name=None, To_name_src='default')

createtextcombined(self, Tt_name=None, Tt_name_src='default', To_name=None, To_name_src='default')

Function: createtext or createtextcombined # Construct a new

Description of Function:

Create a new textcombined secondary method given the the name of the existing texttable and textorientation secondary method to copy the attributes from. If no existing texttable and textorientation secondary method names are given, then the default texttable and textorientation secondary methods will be used as the secondary method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('texttable')
a.show('textorientation')
tc=a.createtextcombined('example1','std','example1','7left')
a.show('texttable')
a.show('textorientation')
```

createtextorientation(self, To_name=None, To_name_src='default')

Function: createtextorientation # Construct a new textorien

Description of Function:

Create a new textorientation secondary method given the the name of the existing textorientation secondary method to copy the attributes from. If no existing textorientation secondary method name is given, then the default textorientation secondary method will be used as the secondary method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('textorientation')
to=a.createtextorientation('example1',)
a.show('textorientation')
to=a.createtextorientation('example2','black')
a.show('textorientation')
```

createtexttable(self, name=None, name_src='default', font=None, spacing=None, expansion=None)

Function: createtexttable # Construct a new textta

Description of Function:

Create a new texttable secondary method given the the name of the existing texttable secondary method to copy the attributes from. If no existing texttable secondary method name is given, then the default texttable secondary method will be used as the secondary method to which the attributes will be copied from.

If the name provided already exists, then a error will be
Secondary method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('texttable')
tt=a.createtexttable('example1',)
a.show('texttable')
tt=a.createtexttable('example2','black')
a.show('texttable')
tt=a.createtexttable(name='new',name_src='red',font=1,space=1,
                    color=242,priority=1,viewport=[0,2.0,0,2.0],
                    worldcoordinate=[0,100,0,50],
                    x=[0,20,40,60,80,100],
                    y=[0,10,20,30,40,50]) # Create instance
a.texttable(tt) # Plot using specific method
```

createvector(self, Gv_name=None, Gv_name_src='default')

Function: createvector # Construct a new vector

Description of Function:

Create a new vector graphics method given the the name and the
vector graphics method to copy the attributes from. If no
vector graphics method name is given, then the default vector
method will be used as the graphics method to which the attributes
be copied from.

If the name provided already exists, then a error will be
method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('vector')
vec=a.createvector('example1',)
a.show('vector')
vec=a.createvector('example2','quick')
a.show('vector')
```

createxvsy(self, GXY_name=None, GXY_name_src='default')

Function: createxvsy # Construct a new XvsY

Description of Function:

Create a new XvsY graphics method given the the name and the
XvsY graphics method to copy the attributes from. If no
XvsY graphics method name is given, then the default XvsY
method will be used as the graphics method to which the attributes
be copied from.

If the name provided already exists, then a error will be
method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('xvsvy')
xy=a.createxvsvy('example1',)
a.show('xvsvy')
xy=a.createxvsvy('example2','quick')
a.show('xvsvy')
```

createxvsvy(self, GXy_name=None, GXy_name_src='default')

Function: createxvsvy

Construct a new Xyvsy

Description of Function:

Create a new Xyvsy graphics method given the the name and Xyvsy graphics method to copy the attributes from. If no Xyvsy graphics method name is given, then the default Xyvsy method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('xyvsvy')
xyy=a.createxyvsvy('example1',)
a.show('xyvsvy')
xyy=a.createxyvsvy('example2','quick')
a.show('xyvsvy')
```

createyxvsvx(self, GYx_name=None, GYx_name_src='default')

Function: createyxvsvx

Construct a new Yxvsvx

Description of Function:

Create a new Yxvsvx graphics method given the the name and Yxvsvx graphics method to copy the attributes from. If no Yxvsvx graphics method name is given, then the default Yxvsvx method will be used as the graphics method to which the attributes will be copied from.

If the name provided already exists, then a error will be raised. Method names must be unique.

Example of Use:

```
a=vcs.init()
a.show('yxvsvx')
yxx=a.createyxvsvx('example1',)
a.show('yxvsvx')
yxx=a.createyxvsvx('example2','quick')
a.show('yxvsvx')
```

destroy(self)

Function: `destroy`

Description of Function:

Destroy the VCS Canvas. It will deallocate the VCS Canvas.

Example of Use:

```
a=vcs.init()
a.plot(array, 'default', 'isofill', 'quick')
a.destory()
```

drawfillarea(self, name=None, style=1, index=1, color=241, priority=1, viewport=[0.0, 1.0, 0.0, 1.0])

Function: `drawfillarea` # Generate and draw a fillarea object on the VCS Canvas.

Description of Function:

Generate and draw a fillarea object on the VCS Canvas.

Example of Use:

```
a=vcs.init()
a.show('fillarea') # Show all the existing fillarea objects
fa=a.drawfillarea(name='red', mtype='dash', size=2,
                  color=242, priority=1, viewport=[0, 2.0, 0, 2.0],
                  worldcoordinate=[0,100, 0,50],
                  x=[0,20,40,60,80,100],
                  y=[0,10,20,30,40,50] ) # Create instance of fillarea
a.fillarea(fa) # Plot using specified fillarea
```

drawline(self, name=None, ltype='solid', width=1, color=241, priority=1, viewport=[0.0, 1.0, 0.0, 1.0])

Function: `drawline` # Generate and draw a line object on the VCS Canvas.

Description of Function:

Generate and draw a line object on the VCS Canvas.

Example of Use:

```
a=vcs.init()
a.show('line') # Show all the existing line objects
ln=a.drawline(name='red', ltype='dash', width=2,
              color=242, priority=1, viewport=[0, 2.0, 0, 2.0],
              worldcoordinate=[0,100, 0,50],
              x=[0,20,40,60,80,100],
              y=[0,10,20,30,40,50] ) # Create instance of line
a.line(ln) # Plot using specified line
```

drawmarker(self, name=None, mtype='solid', size=1, color=241, priority=1, viewport=[0.0, 1.0, 0.0, 1.0])

Function: `drawmarker` # Generate and draw a marker object on the VCS Canvas.

Description of Function:

Generate and draw a marker object on the VCS Canvas.

Example of Use:

```
a=vcs.init()
a.show('marker') # Show all the existing marker objects
mrk=a.drawmarker(name='red', mtype='dash', size=2,
```

```

        color=242, priority=1, viewport=[0, 2.0, 0,
worldcoordinate=[0,100, 0,50]
x=[0,20,40,60,80,100],
y=[0,10,20,30,40,50] )      # Create instance
a.marker(mrk)                # Plot using specific

```

drawtext = drawtextcombined(self, Tt_name=None, To_name=None, string=None, font=1, spacing=1)

drawtextcombined(self, Tt_name=None, To_name=None, string=None, font=1, spacing=2, expands=1)

Function: drawtexttable # Generate and draw a texttable

Description of Function:

Generate and draw a texttable object on the VCS Canvas.

Example of Use:

```

a=vcs.init()
a.show('texttable')          # Show all the text
tt=a.drawtexttable(Tt_name = 'red', To_name='7left', mtype='t',
                    color=242, priority=1, viewport=[0, 2.0, 0,
worldcoordinate=[0,100, 0,50]
x=[0,20,40,60,80,100],
y=[0,10,20,30,40,50] )      # Create instance
a.texttable(tt)              # Plot using specific

```

eps(self, *args)

Function: Encapsulated PostScript

Description of Function:

In some cases, the user may want to save the plot out as a PostScript image. This routine allows the user to save the plot as an Encapsulated PostScript file. This file can be converted to other image formats with the various such imaging tools found freely on the web.

If no path/file name is given and no previously created gif file is designated, then file

/\$HOME/PCMDI_GRAPHICS/default.eps

will be used for storing gif images. However, if a previously designated file is designated, that file will be used for gif output.

By default, the page orientation is in Landscape mode (l). To change orientation to portrait mode (p), enter 'p' as the second argument.

The eps command is used to create or append to a gif file. The arguments for saving a eps file: `Append' mode (a) appends gif output to the file; `Replace' (r) mode overwrites an existing eps file with the new. The default mode is to overwrite an existing eps file (i.e. mode 'r').

Example of Use:

```

a=vcs.init()

```

```

a.plot(array)
a.eps('example')           # overwrite an existing eps file
a.eps('example', 'a')      # append to an existing eps file
a.eps('example','r')       # overwrite existing eps file
a.eps('example','r','p')   # overwrite eps image file with n

```

fillarea(self, *args, **parms)

Function: fillarea # Generate a fill

Description of Function:

Plot a fillarea segment on the Vcs Canvas. If no fillarea object is given, then an error will be returned.

Example of Use:

```

a=vcs.init()
a.show('fillarea')           # Show all the existin
fa=a.getfillarea('red')      # Create instance of '
fa.style=1                    # Set the fillarea sty
fa.index=4                    # Set the fillarea inc
fa.color = 242                # Set the fillarea col
fa.type = 4                    # Set the fillarea typ
fa.x=[[0.0,2.0,2.0,0.0,0.0], [0.5,1.5]] # Set the x value
fa.y=[[0.0,0.0,2.0,2.0,0.0], [1.0,1.0]] # Set the y value
a.fillarea(fa)                # Plot using speci

```

flush(self, *args)

Function: flush

Description of Function:

The flush command executes all buffered X events in the qu

Example of Use:

```

a=vcs.init()
a.plot(array,'default','isofill','quick')
a.flush()

```

generate_gm(self, type, name)

geometry(self, *args)

Function: geometry

Description of Function:

The geometry command is used to set the size and position

Example of Use:

```

a=vcs.init()
a.plot(array,'default','isofill','quick')
a.geometry(450,337)

```

get_gm(self, type, name)

getboxfill(self, Gfb_name_src='default')

```
Function: getboxfill # Construct a new
```

Description of Function:

VCS contains a list of graphics methods. This function will
boxfill class object from an existing VCS boxfill graphics
no boxfill name is given, then boxfill 'default' will be u

Note, VCS does not allow the modification of `default' attr
sets. However, a `default' attribute set that has been cop
different name can be modified. (See the createboxfill fun

Example of Use:

```
a=vcs.init()
a.show('boxfill') # Show all the existin
box=a.getboxfill() # box instance of 'def
# method
box2=a.getboxfill('quick') # box2 instance of exi
# graphics met
```

getcolorcell(self, *args)

Function: getcolorcell

Description of Function:

Get an individual color cell in the active colormap. If de
the active colormap, then return an error string.

If the the visul display is 16-bit, 24-bit, or 32-bit True
of the VCS Canvas is made evertime the color cell is chang

Note, the user can only change color cells 0 through 239 a
value must range from 0 to 100. Where 0 represents no col
and 100 is the greatest color intensity.

Example of Use:

```
a=vcs.init()
a.plot(array, 'default', 'isofill', 'quick')
a.setcolormap("AMIP")
a.getcolorcell(11, 0, 0, 0)
a.getcolorcell(21, 100, 0, 0)
a.getcolorcell(31, 0, 100, 0)
a.getcolorcell(41, 0, 0, 100)
a.getcolorcell(51, 100, 100, 100)
a.getcolorcell(61, 70, 70, 70)
```

getcolormap(self, Cp_name_src='default')

Function: getcolormap # Construct a new

Description of Function:

VCS contains a list of secondary methods. This function wi
colormap class object from an existing VCS colormap second
no colormap name is given, then colormap 'default' will be

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the `createcolormap` function.)

Example of Use:

```
a=vcs.init()
a.show('colormap')           # Show all the existing
cp=a.getcolormap()           # cp instance of colormap
                                # colormap method
cp2=a.getcolormap('quick')    # cp2 instance of colormap
                                # secondary colormap
```

getcolormapname(self, *args)

Function: `getcolormapcell`

Description of Function:

Get colormap name of the active colormap.

Example of Use:

```
a=vcs.init()
a.plot(array, 'default', 'isofill', 'quick')
a.getcolormapname()
```

getcontinents(self, Gcon_name_src='default')

Function: `getcontinents` # Construct a new continents class object

Description of Function:

VCS contains a list of graphics methods. This function will return a continents class object from an existing VCS continents graphics method. If no continents name is given, then continents 'default' will be used.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the `createcontinents` function.)

Example of Use:

```
a=vcs.init()
a.show('continents')         # Show all the existing
                                # continents graphics methods
con=a.getcontinents()        # con instance of 'default'
                                # continents graphics method
con2=a.getcontinents('quick') # con2 instance of existing
                                # continents graphics method
```

getcontinentstype(self, *args)

Function: `getcontinentstype`

Description of Function:

Retrieve continents type from VCS. Remember the value can be 0 and 11.

Example of Use:

```
a=vcs.init()
cont_type = a.getcontinentstype() # Get the continents type
```

getfillarea(self, name='default', style=None, index=None, color=None, priority=None, viewport=None)

Function: getfillarea # Construct a new fillarea

Description of Function:

VCS contains a list of secondary methods. This function will create a fillarea class object from an existing VCS fillarea secondary method. If no fillarea name is given, then fillarea 'default' will be used.

Note, VCS does not allow the modification of 'default' attributes. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createfillarea function)

Example of Use:

```
a=vcs.init()
a.show('fillarea') # Show all the existing fillarea
fa=a.getfillarea() # fa instance of 'default' fillarea
# method
fa2=a.getfillarea('quick') # fa2 instance of existing fillarea
# secondary method
fa3=a.createmarker(name='new', name='red', style=1, index=1,
                  color=242, priority=1, viewport=[0, 2.0, 0, 2.0],
                  worldcoordinate=[0,100, 0,50],
                  x=[0,20,40,60,80,100],
                  y=[0,10,20,30,40,50] ) # Create instance of fillarea
a.fillarea(fa3) # Plot using specified fillarea
```

getisofill(self, Gfi_name_src='default')

Function: getisofill Construct a new isofill graphics

Description of Function:

VCS contains a list of graphics methods. This function will create an isofill class object from an existing VCS isofill graphics secondary method. If no isofill name is given, then isofill 'default' will be used.

Note, VCS does not allow the modification of 'default' attributes. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createisofill function)

Example of Use:

```
a=vcs.init()
a.show('isofill') # Show all the existing isofill
iso=a.getisofill() # iso instance of 'default' isofill
# method
iso2=a.getisofill('quick') # iso2 instance of existing isofill
# graphics method
```

getisoline(self, Gi_name_src='default')

```
Function: getisoline # Construct a new
```

Description of Function:

VCS contains a list of graphics methods. This function will return an isoline class object from an existing VCS isoline graphics method. If no isoline name is given, then isoline 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createisoline function.)

Example of Use:

```
a=vcs.init()
a.show('isoline') # Show all the existing isoline
iso=a.getisoline() # iso instance of 'default'
# method
iso2=a.getisoline('quick') # iso2 instance of existing
# graphics method
```

getline(self, name='default', ltype=None, width=None, color=None, priority=None, viewport=None)

```
Function: getline # Construct a new line secondary method
```

Description of Function:

VCS contains a list of secondary methods. This function will return a line class object from an existing VCS line secondary method. If no line name is given, then line 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createline function.)

Example of Use:

```
a=vcs.init()
a.show('line') # Show all the existing line
ln=a.getline() # ln instance of 'default'
# method
ln2=a.getline('quick') # ln2 instance of existing
# secondary method
ln3=a.getline(name='red', ltype='dash', width=2,
              color=242, priority=1, viewport=[0, 2.0, 0,
              worldcoordinate=[0,100, 0,50]
              x=[0,20,40,60,80,100],
              y=[0,10,20,30,40,50] ) # Create instance
a.line(ln3) # Plot using specified line
```

getmarker(self, name='default', mtype=None, size=None, color=None, priority=None, viewport=None)

```
Function: getmarker # Construct a new marker
```

Description of Function:

VCS contains a list of secondary methods. This function will return a marker class object from an existing VCS marker secondary method. If no marker name is given, then marker 'default' will be used.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the createmarker function)

Example of Use:

```
a=vcs.init()
a.show('marker') # Show all the existing
mrk=a.getmarker() # mrk instance of 'default'
# method
mrk2=a.getmarker('quick') # mrk2 instance of existing
# secondary method
mrk3=a.getmarker(name='red', mtype='dash', size=2,
                  color=242, priority=1, viewport=[0, 2.0, 0, 2.0],
                  worldcoordinate=[0,100, 0,50],
                  x=[0,20,40,60,80,100],
                  y=[0,10,20,30,40,50] ) # Create instance
a.marker(mrk3) # Plot using specified method
```

getmeshfill(self, Gfm_name_src='default')

Function: getmeshfill # Construct a new meshfill

Description of Function:

VCS contains a list of graphics methods. This function will return a meshfill class object from an existing VCS meshfill graphics method. If no meshfill name is given, then meshfill 'default' will be used.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the createmeshfill function)

Example of Use:

```
a=vcs.init()
a.show('meshfill') # Show all the existing
mesh=a.getmeshfill() # mesh instance of 'default'
# method
mesh2=a.getmeshfill('quick') # mesh2 instance of existing
# graphics method
```

getoutfill(self, Gfo_name_src='default')

Function: getoutfill # Construct a new outfill

Description of Function:

VCS contains a list of graphics methods. This function will return an outfill class object from an existing VCS outfill graphics method. If no outfill name is given, then outfill 'default' will be used.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the createoutfill function)

Example of Use:

```

a=vcs.init()
a.show('outfill') # Show all the existing
out=a.getoutfill() # out instance of 'default'
# method
out2=a.getoutfill('quick') # out2 instance of existing
# graphics methods

```

getoutline(self, Go_name_src='default')

Function: getoutline # Construct a new

Description of Function:

VCS contains a list of graphics methods. This function will create an outline class object from an existing VCS outline graphics method. If no outline name is given, then outline 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied with a different name can be modified. (See the createoutline function)

Example of Use:

```

a=vcs.init()
a.show('outline') # Show all the existing
out=a.getoutline() # out instance of 'default'
# method
out2=a.getoutline('quick') # out2 instance of existing
# graphics methods

```

getplot(self, Dp_name_src='default', template=None)

Function: getplot # Get existing display plot

Description of Function:

This function will create a display plot object from an existing VCS plot object. If no display plot name is given, then None is returned.

Example of Use:

```

a=vcs.init()
a.show('template') # Show all the existing
plot1=a.getplot('dpy_plot_1') # plot1 instance of 'default'

```

getprojection(self, Proj_name_src='default')

Function: getprojection # Construct a new

Description of Function:

VCS contains a list of graphics methods. This function will create a projection class object from an existing VCS projection method. If no projection name is given, then projection 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied with a different name can be modified. (See the createprojection function)

Example of Use:

```
a=vcs.init()
a.show('projection') # Show all the exist
box=a.getprojection() # box instance of '
# method
box2=a.getprojection('quick') # box2 instance of
# graphics met
```

getscatter(self, GSp_name_src='default')

```
Function: getscatter # Construct a new scat
```

Description of Function:

VCS contains a list of graphics methods. This function will
scatter class object from an existing VCS scatter graphics
no scatter name is given, then scatter 'default' will be u

Note, VCS does not allow the modification of `default' att
sets. However, a `default' attribute set that has been cop
different name can be modified. (See the createscatter fun

Example of Use:

```
a=vcs.init()
a.show('scatter') # Show all the existin
sct=a.getscatter() # sct instance of 'def
# method
sct2=a.getscatter('quick') # sct2 instance of exi
# graphics metho
```

gettaylordiagram(self, Gtd_name_src='default')

```
Function: gettayloridigram # Construct a
```

Description of Function:

VCS contains a list of graphics methods. This function will
taylordiagram class object from an existing VCS taylordiag
no taylordiagram name is given, then taylordiagram 'default

Note, VCS does not allow the modification of `default' att
sets. However, a `default' attribute set that has been cop
different name can be modified. (See the createboxfill fun

Example of Use:

```
a=vcs.init()
a.show('taylordiagram') # Show all the
td=a.gettaylordiagram() # td instance o
# method
td2=a.gettaylordiagram('default') # td2 instance
# graph
```

gettemplate(self, Pt_name_src='default')

```
Function: gettemplate # Construct a new
```

Description of Function:

VCS contains a list of predefined templates. This function returns a template class object from an existing VCS template. If no name is given, then template 'default' will be used.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the createtemplate function)

Example of Use:

```
a=vcs.init()
a.show('template')           # Show all the existing templates
templt=a.gettemplate()       # templt instance of a template
templt2=a.gettemplate('quick') # templt2 contains 'quick' template
```

gettext = gettextcombined(self, Tt_name_src='default', To_name_src='default', string=None, font=None, spacing=None)

gettextcombined(self, Tt_name_src='default', To_name_src='default', string=None, font=None, spacing=None)

Function: gettext or gettextcombined # Construct a new textcombined class object

Description of Function:

VCS contains a list of secondary methods. This function will return a textcombined class object from an existing VCS texttable secondary method and an existing VCS textorientation secondary method. If no texttable or textorientation names are given, then the 'default' will be used in both cases.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the createtextcombined function)

Example of Use:

```
a=vcs.init()
a.show('texttable')         # Show all the existing texttables
a.show('textorientation')   # Show all the existing textorientations
tc=a.gettextcombined()      # Use 'default' for texttable and textorientation
tc2=a.gettextcombined('std','7left') # Use 'std' texttable and '7left' textorientation
if istextcombined(tc):      # Check to see if textcombined
    tc.list()               # Print out all its attributes
```

gettextorientation(self, To_name_src='default')

Function: gettextorientation # Construct a new textorientation class object

Description of Function:

VCS contains a list of secondary methods. This function will return a textorientation class object from an existing VCS textorientation secondary method. If no textorientation name is given, then the textorientation 'default' will be used.

Note, VCS does not allow the modification of `default` attribute sets. However, a `default` attribute set that has been copied under a different name can be modified. (See the createtextorientation function)

Example of Use:

```
a=vcs.init()
a.show('textorientation')      # Show all the existing text
to=a.gettextorientation()      # to instance of 'default' te
                                #          method
to2=a.gettextorientation('quick') # to2 instance of exist
                                #          secondary metho
```

gettexttable(self, name='default', font=None, spacing=None, expansion=None, color=None, priority=None)
Function: gettexttable # Construct a new texttable

Description of Function:

VCS contains a list of secondary methods. This function will create a texttable class object from an existing VCS texttable secondary method. If no texttable name is given, then texttable 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createtexttable function for more details.)

Example of Use:

```
a=vcs.init()
a.show('texttable')            # Show all the existing t
tt=a.gettexttable()            # tt instance of 'default'
                                #          method
tt2=a.gettexttable('quick')    # tt2 instance of existin
                                #          secondary method
tt3=a.gettexttable(name='red', font=1, spacing=1, expansion=1,
                                color=242, priority=1, viewport=[0, 2.0, 0,
                                worldcoordinate=[0,100, 0,50]
                                x=[0,20,40,60,80,100],
                                y=[0,10,20,30,40,50] )      # Create instance
a.texttable(tt3)                # Plot using specific
```

getvector(self, Gv_name_src='default')
Function: getvector # Construct a new vector

Description of Function:

VCS contains a list of graphics methods. This function will create a vector class object from an existing VCS vector graphics method. If no vector name is given, then vector 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createvector function for more details.)

Example of Use:

```
a=vcs.init()
a.show('vector')              # Show all the existing
vec=a.getvector()              # vec instance of 'defa
                                #          method
vec2=a.getvector('quick')     # vec2 instance of exis
```

graphics method

getxvsy(self, GXY_name_src='default')

Function: getxvsy # Construct a new XvsY graphics method

Description of Function:

VCS contains a list of graphics methods. This function will return an XvsY class object from an existing VCS XvsY graphics method. If no XvsY name is given, then XvsY 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied with a different name can be modified. (See the createxvsy function)

Example of Use:

```
a=vcs.init()
a.show('xvsy') # Show all the existing XvsY graphics methods
xy=a.getxvsy() # xy instance of 'default' XvsY graphics method
xy2=a.getxvsy('quick') # xy2 instance of existing XvsY graphics method
```

getxyvsy(self, GXy_name_src='default')

Function: getxyvsy # Construct a new Xyvsy graphics method

Description of Function:

VCS contains a list of graphics methods. This function will return an Xyvsy class object from an existing VCS Xyvsy graphics method. If no Xyvsy name is given, then Xyvsy 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied with a different name can be modified. (See the createxyvsy function)

Example of Use:

```
a=vcs.init()
a.show('xyvsy') # Show all the existing Xyvsy graphics methods
xyy=a.getxyvsy() # xyy instance of 'default' Xyvsy graphics method
xyy2=a.getxyvsy('quick') # xyy2 instance of existing Xyvsy graphics method
```

getyvsx(self, GYx_name_src='default')

Function: getyvsx # Construct a new Yxvsx graphics method

Description of Function:

VCS contains a list of graphics methods. This function will return a Yxvsx class object from an existing VCS Yxvsx graphics method. If no Yxvsx name is given, then Yxvsx 'default' will be used.

Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied with a different name can be modified. (See the createyvsx function)

different name can be modified. (See the `createyxvsx` function)

Example of Use:

```
a=vcs.init()
a.show('yxvsx') # Show all the existing
yxx=a.getyxvsx() # yxx instance of 'def
# method
yxx2=a.getyxvsx('quick') # yxx2 instance of exi
# graphics method
```

gif(self, filename='noname.gif', merge='r', orientation='l', geometry='792x612')

Function: gif

Description of Function:

In some cases, the user may want to save the plot out as a gif file. This routine allows the user to save the VCS canvas output as a gif file. This file can be converted to other gif formats with the use of such imaging tools found freely on the web.

If no path/file name is given and no previously created gif file is designated, then file

```
/$HOME/PCMDI_GRAPHICS/default.gif
```

will be used for storing gif images. However, if a previously designated file is designated, that file will be used for gif output.

By default, the page orientation is in Landscape mode (l). To change the orientation to portrait mode (p), set the orientation = 'p'.

The GIF command is used to create or append to a gif file. The modes for saving a gif file: 'Append' mode (a) appends gif output to the file; 'Replace' (r) mode overwrites an existing gif file with the new file. The default mode is to overwrite an existing gif file (i.e. 'r').

Example of Use:

```
a=vcs.init()
a.plot(array)
a.gif(filename='example.gif', merge='a', orientation='l',
a.gif('example') # overwrite existing gif file (def
a.gif('example',merge='r') # overwrite existing gif file
a.gif('example',merge='a') # merge gif image into exist
a.gif('example',orientation='l') # merge gif image into ex
a.gif('example',orientation='p') # merge gif image into ex
a.gif('example',geometry='600x500') # merge gif image into
```

graphicsmethodgui(self, gm_type='boxfill', gm_name='default', gui_parent=None)

Function: graphicsmethodgui

Description of Function:

Run the VCS graphicsmethod interface.

The `graphicsmethodgui` command is used to bring up the VCS
The interface is used to alter existing graphics method at

Example of Use:

```
a=vcs.init()  
a.graphicsmethodgui('boxfill', 'quick')
```

grid(self, *args)

Function: `grid`

Description of Function:

Set the default plotting region for variables that have mo
than the graphics method. This will also be used for anima
third and fourth dimensions.

Example of Use:

```
a=vcs.init()  
a.grid(12,12,0,71,0,45)
```

gs(self, filename='noname.gs', device='png256', orientation='l', resolution='792x612')

Function: `gs`

Description of Function:

This routine allows the user to save the VCS canvas in one
GhostScript (`gs`) file types (also known as devices). To vi
GhostScript devices, issue the command "`gs --help`" at the
prompt. Device names include: `bmp256`, `epswrite`, `jpeg`, `jpeg`
`pdfwrite`, `png256`, `png16m`, `sgirgb`, `tiffpack`, and `tifflzw`. E
the device = '`png256`'.

If no path/file name is given and no previously created `gs`
designated, then file

```
/$HOME/PCMDI_GRAPHICS/default.gs
```

will be used for storing `gs` images. However, if a previous
file exist, then this output file will be used for storage

By default, the page orientation is in Landscape mode (`l`).
the page orientation to portrait mode (`p`), set the paramet

The `gs` command is used to create a single `gs` file at this
can use other tools to append separate image files.

Example of Use:

```
a=vcs.init()  
a.plot(array)  
a.gs('example') #defaults: device='png256', orientation='l'  
a.gs(filename='example.tif', device='tiffpack', orientatio  
a.gs(filename='example.pdf', device='pdfwrite', orientatio  
a.gs(filename='example.jpg', device='jpeg', orientation='p
```

iscanvasdisplayed(self, *args)

Function: `iscanvasdisplayed` # Return 1 if a VCS Canvas

Description of Function:

This function returns a 1 if a VCS Canvas is displayed or no VCS Canvas is displayed on the screen.

Example of Use:

```
a=vcs.init()
...

a.iscanvasdisplayed()
```

isinfile(self, GM, file=None)

Checks if a graphic method is stored in a file
if no file name is passed then looks into the initial.attribu

islandscape(self)

Function: `islandscape`

Description of Function:

Indicates if VCS's orientation is landscape.

Returns a 1 if orientation is landscape.

Otherwise, it will return a 0, indicating false (not in la

Example of Use:

```
a=vcs.init()
...

if a.islandscape():
    a.portrait() # Set VCS's orientation to p
```

isofill(self, *args, **parms)

Function: `isofill` # Generate an isofill

Description of Function:

Generate a isofill plot given the data, isofill graphics m
template. If no isofill class object is given, then the 'c
graphics method is used. Similarly, if no template class c
then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('isofill') # Show all the existin
iso=a.getisofill('quick') # Create instance of '
a.isofill(array,iso) # Plot array using spe
# template
a.clear() # Clear VCS canvas
a.isofill(array,iso,template) # Plot array using spe
```

isoline(self, *args, **parms)

Function: `isoline` # Generate an isoline

Description of Function:

Generate a isoline plot given the data, isoline graphics method and a template. If no isoline class object is given, then the 'default' graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('isoline') # Show all the existing isolines
iso=a.getisoline('quick') # Create instance of 'quick'
a.isoline(array,iso) # Plot array using specified
# template
a.clear() # Clear VCS canvas
a.isoline(array,iso,template) # Plot array using specified
```

isportrait(self)

Function: `isportrait`

Description of Function:

Indicates if VCS's orientation is portrait.

Returns a 1 if orientation is portrait.

Otherwise, it will return a 0, indicating false (not in portrait).

Example of Use:

```
a=vcs.init()
...

if a.isportrait():
    a.landscape() # Set VCS's orientation to landscape
```

landscape(self, *args)

Function: `landscape`

Description of Function:

Change the VCS Canvas orientation to Landscape.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.landscape() # Change the VCS Canvas orientation and set
```

line(self, *args, **parms)

Function: `line` # Generate a line plot

Description of Function:

Plot a line segment on the Vcs Canvas. If no line class object is given, then an error will be returned.

Example of Use:

```

a=vcs.init()
a.show('line') # Show all the existing
ln=a.getline('red') # Create instance of
ln.width=4 # Set the line width
ln.color = 242 # Set the line color
ln.type = 4 # Set the line type
ln.x=[[0.0,2.0,2.0,0.0,0.0], [0.5,1.5]] # Set the x value
ln.y=[[0.0,0.0,2.0,2.0,0.0], [1.0,1.0]] # Set the y value
a.line(ln) # Plot using specified

```

listelements(self, *args)

Function: listelements

Description of Function:

Returns a Python list of all the VCS class objects.

The list that will be returned:

```

['template', 'boxfill', 'continent', 'isofill', 'isoline',
 'scatter', 'vector', 'xvsy', 'xyvsy', 'yxvsx', 'colormap',
 'line', 'list', 'marker', 'text']

```

Example of Use:

```

a=vcs.init()
a.listelements()

```

marker(self, *args, **parms)

Function: marker # Generate a marker

Description of Function:

Plot a marker segment on the Vcs Canvas. If no marker class object is given, then an error will be returned.

Example of Use:

```

a=vcs.init()
a.show('marker') # Show all the existing
mrk=a.getmarker('red') # Create instance of
mrk.size=4 # Set the marker size
mrk.color = 242 # Set the marker color
mrk.type = 4 # Set the marker type
mrk.x=[[0.0,2.0,2.0,0.0,0.0], [0.5,1.5]] # Set the x value
mrk.y=[[0.0,0.0,2.0,2.0,0.0], [1.0,1.0]] # Set the y value
a.marker(mrk) # Plot using specified

```

match_color(self, color, colormap=None)

Function: cmatch_color # Returns the

Description of Function:

Given a color (defined as rgb values -0/100 range)-
returns the color number that is closest from the map
(using rms difference between rgb values)
if colormap is not map use the currently used colormap

Example of use:

```

a=vcs.init()

```

```
print a.match_color('salmon')
print a.match_color('red')
print a.match_color([0,0,100], 'default') # closes
```

meshfill(self, *args, **parms)

Function: meshfill # Generate an meshfill plot

Description of Function:

Generate a meshfill plot given the data, the mesh, a meshfill class object, and a template. If no meshfill class object is given, then the `Meshfill` class object is used. Similarly, if no template class object is given, then the 'default' template is used.

Format:

This function expects 1D data (any extra dimension will be ignored). In addition the mesh array must be of the same shape than the data array. Let's say you want to plot a spatial data assuming a mesh containing 10000 cells. The shape of the mesh, assuming 4 vertices per grid cell, in brief you'd have:
data.shape=(10000,)
mesh.shape=(10000,2,4)

Example of Use:

```
a=vcs.init()
a.show('meshfill') # Show all the existing meshfill plots
mesh=a.getmeshfill() # Create instance of Meshfill class
a.meshfill(array,mesh) # Plot array using spatial data
# template
a.clear() # Clear VCS canvas
a.meshfill(array,mesh,mesh_graphic_method,template) # Plot
```

objecthelp(self, *arg)

Function: objecthelp # Print out the object's class

Description of Function:

Print out information on the VCS object. See example below

Example of Use:

```
a=vcs.init()

ln=a.getline('red') # Get a VCS line object
a.objecthelp(ln) # This will print out
```

open(self, *args)

Function: open

Description of Function:

Open VCS Canvas object. This routine really just manages to popup the VCS Canvas for viewing. It can be used to display

Example of Use:

```
a=vcs.init()
```

```
a.open()
```

orientation(self, *args)

Function: orientation

Description of Function:

Return VCS's orientation. Will return either Portrait or Landscape

Example of Use:

```
a=vcs.init()
a.orientation()          # Return either "landscape" or "portrait"
```

outfill(self, *args, **parms)

Function: outfill # Generate an outfill plot

Description of Function:

Generate a outfill plot given the data, outfill graphics method, and template. If no outfill class object is given, then the 'default' graphics method is used. Simerly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('outfill')          # Show all the existing plots
out=a.getoutfill('quick')  # Create instance of outfill class
a.outfill(array,out)       # Plot array using specified
                           # template
a.clear()                  # Clear VCS canvas
a.outfill(array,out,template) # Plot array using specified template
```

outline(self, *args, **parms)

Function: outline # Generate an outline plot

Description of Function:

Generate a outline plot given the data, outline graphics method, and template. If no outline class object is given, then the 'default' graphics method is used. Simerly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('outline')          # Show all the existing plots
out=a.getoutline('quick')  # Create instance of outline class
a.outline(array,out)       # Plot array using specified
                           # template
a.clear()                  # Clear VCS canvas
a.outline(array,out,template) # Plot array using specified template
```

page(self, *args)

Function: page

Description of Function:

Change the VCS Canvas orientation to either 'portrait' or

The orientation of the VCS Canvas and of cgm and raster in the PAGE command. Only portrait (y > x) or landscape (x > y) is permitted.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.page()      # Change the VCS Canvas orientation and set
```

pageeditor(self, gui_parent=None, continents=None)

Function: pageeditor

Description of Function:

Run the VCS page editor GUI.

The pageeditor command is used to bring up the VCS page editor GUI. The interface is used to design canvases.

Example of Use:

```
a=vcs.init()
a.pageeditor()
```

pdf(self, *args)

Function: pdf

Description of Function:

To save out a postscript file, VCS will first create a cgm file in the user's PCMDI_GRAPHICS directory. Then use gplot to convert the cgm file to a postscript file in the user's chosen directory. And then convert it pdf using ps2pdf.

The pdf command is used to create a pdf file. Orientation is either 'l' = landscape or 'p' = portrait. The default is landscape.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.pdf('example')      # Creates a landscape pdf file
a.pdf('example','p')  # Creates a portrait pdf file
```

plot(self, *actual_args, **keyargs)

Function: plot

Description of plot:

Plot an array(s) of data given a template and graphics method. The method is used to define where the data and variable attributes will be plotted on the Canvas. The VCS graphics method is used to define how the data is plotted on the VCS Canvas.

The form of the call is:

```
plot(array1=None, array2=None, template_name=None, graphics_name=None, [key=value [, key=value [, ...
```

where array1 and array2 are NumPy arrays.

Plot keywords:

```
ratio [default is none]
None: let the self.ratio attribute decide
0,'off': overwrite self.ratio and do nothing about it
'auto': computes an automatic ratio
'3',3: y dim will be 3 times bigger than x dim (relative to x)
Adding a 't' at the end of the ratio, makes the time axis
```

Variable attribute keys:

```
comment1      = string      #Comment plotted above file
comment2      = string      #Comment plotted above comment
comment3      = string      #Comment plotted above comment
comment4      = string      #Comment plotted above comment
file_comment  = string      #Comment (defaults to file_name)
hms           = string      (hh:mm:ss) #Hour, minute, second
long_name     = string      #Descriptive variable name
name          = string      #Variable name (defaults to file_name)
time         = cftime      #instance (relative or absolute)
                    cftime, reltime or abstime
units        = string      #Variable units
ymd          = string      (yy/mm/dd) #Year, month, day
```

Dimension attribute keys (dimension length=n):

```
[x|y|z|t|w]array1 = NumPy array of length n      # x or y
[x|y|z|t|w]array2 = NumPy array of length n      # x or y
[x|y]bounds       = NumPy array of shape (n,2)   # x or y
[x|y|z|t|w]name   = string                      # x or y
[x|y|z|t|w]units  = string                      # x or y
[x|y]weights      = NumPy array of length n      # x or y
                    calcul
```

CDMS object:

```
[x|y|z|t|w]axis  = CDMS axis object             # x or y
grid             = CDMS grid object             # Grid object
variable        = CDMS variable object         # Variable object
```

Other:

```
[x|y]rev         = 0|1                          # if ==1
                    c
contints         = 0,1,2,3,4,5,6,7,8,9,10,11    # if >=1
                    (
                    1
                    x
                    ,
                    # The co
                    # rangin
                    # 0 s
```

```
# 1 s
# 2 s
# 3 s
# 4 s
# 5 s

# Values
# define
# throug
```

Graphics Output in Background Mode:

```
bg = 0|1 # if ==1, create images in t
(
```

Note:

More specific attributes take precedence over general attributes. More specific attributes override variable object attributes, data arrays override axis objects, which override grid objects, and so on.

For example, if both 'file_comment' and 'variable' keywords are used, 'file_comment' is used instead of the file comment in the plot title. If both 'xaxis' and 'grid' keywords are specified, the value of 'xaxis' overrides the x-axis of grid.

Example of Use:

```
x=vcs.init() # x is an instance of the VCS class object
x.plot(array) # this call will use default settings
x.plot(array, 'AMIP', 'isofill', 'AMIP_psl') # this is specific
graphics method
t=x.gettemplate('AMIP') # get a predefined the template
vec=x.getvector('quick') # get a predefined the vector
x.plot(array1, array2, t, vec) # plot the data as a vector
x.clear() # clear the VCS Canvas of a
box=x.createboxfill('new') # create boxfill graphics method
x.plot(box,t,array) # plot array data using boxfill
```

plot_filledcontinents(self, slab, template_name, g_type, g_name, bg, ratio)

portrait(self, *args)

Function: portrait

Description of Function:

Change the VCS Canvas orientation to Portrait.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.portrait() # Change the VCS Canvas orientation and
```

postscript(self, *args)

Function: `postscript`

Description of Function:

Postscript output is another form of vector graphics. It is the counterpart, because it is stored out in ASCII format. To use VCS will first create a cgm file in the user's PCMDI_GRAPHICS directory. To use `gplot` to convert the cgm file to a postscript file in a chosen directory.

There are two modes for saving a postscript file: 'Append' (a) to append output to an existing postscript file; and 'Replace' (r) to replace an existing postscript file with new postscript output. The default mode is 'Append' (a) to the existing postscript file (i.e. mode (r)).

The `POSTSCRIPT` command is used to create a postscript file in landscape or 'p' = portrait. The default is landscape.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.postscript('example')           # Overwrite a landscape postscript
a.postscript('example','a')       # Append postscript to an existing
a.postscript('example','r')       # Overwrite an existing file
a.postscript('example','r','p')   # Overwrite postscript in portrait
```

printer(self, *args)

Function: `printer`

Description of Function:

This function creates a temporary cgm file and then sends it to the printer. Once the printer received the information, then the cgm file is deleted. The temporary cgm file is created in the user's PCMDI_GRAPHICS directory.

The `PRINTER` command is used to send the VCS Canvas plot(s) to the printer. Orientation can be either: 'l' = landscape, or 'p' = portrait.

Note: VCS graphical displays can be printed only if the user has the `lpr` file (included with the VCS software) for the home system. The `lpr` file must be:

```
/$HOME/PCMDI_GRAPHICS/HARD_COPY
```

where `/$HOME` denotes the user's home directory.

For more information on the `HARD_COPY` file, see URL:

http://www-pcmdi.llnl.gov/software/vcs/vcs_guidetoc.html#1

Example of Use:

```
a=vcs.init()
a.plot(array)
```

```
a.printer('printer_name') # Send plot(s) to postscript printer
```

projectiongui(self, gui_parent=None, projection='default')

Function: projectiongui

Description of Function:

Run the VCS projection editor interface.

The projectiongui command is used to bring up the VCS projection editor interface. This command is used to select, create, change, or remove projections.

Example of Use:

```
a=vcs.init()
a.projectiongui()
```

pstogif(self, filename, *opt)

Function: pstogif

Description of Function:

In some cases, the user may want to save the plot out as a postscript file. This routine allows the user to convert a postscript file to a gif or pdf file.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.pstogif('filename.ps') # convert the postscript file to a gif file
a.pstogif('filename.ps','l') # convert the postscript file to a pdf file
a.pstogif('filename.ps','p') # convert the postscript file to a pdf file
```

raster(self, *args)

Function: raster

Description of Function:

In some cases, the user may want to save the plot out as a raster image. This routine allows the user to save the VCS canvas output as a raster image. This file can be converted to other raster formats with tools such as ImageMagick or Ghostscript. Such imaging tools are found freely on the web.

If no path/file name is given and no previously created raster file is designated, then file

```
/$HOME/PCMDI_GRAPHICS/default.ras
```

will be used for storing raster images. However, if a previously designated file is designated, that file will be used for raster output.

Example of Use:

```
a=vcs.init()
a.plot(array)
a.raster('example','a') # append raster image to existing file
a.raster('example','r') # overwrite existing raster file
```

removeP(self, *args)

remove_display_name(self, *args)

```
#####  
#  
# VCS utility wrapper to remove the display names.  
#  
#####
```

removeobject(self, obj)

Function: remove

Description of Function:

The user has the ability to create primary and secondary objects. The function allows the user to remove these objects from the appropriate class list.

Note, To remove the object completely from Python, remember use the "del" function.

Also note, The user is not allowed to remove a "default" object.

Example of Use:

```
a=vcs.init()  
line=a.getline('red') # To Modify an existing line o  
iso=x.createisoline('dean') # Create an instance of an iso  
...  
x.remove(line) # Removes line object from VCS list  
del line # Destroy instance "line", garbage coll  
x.remove(iso) # Remove isoline object from VCS list  
del iso # Destroy instance "iso", garbage coll
```

resetgrid(self, *args)

Function: resetgrid

Description of Function:

Set the plotting region to default values.

Example of Use:

Not Working!

return_display_ON_num(self, *args)

```
#####  
#  
# VCS utility wrapper to return the number of displays that a  
#  
#####
```

return_display_names(self, *args)

```
#####  
#
```

```
# VCS utility wrapper to return the current display names.
#
#####
```

saveinitialfile(self)

Function: `saveinitialfile` # Save initial

Description of Function:

At start-up, VCS reads a script file named `initial.attributes` defines the initial appearance of the VCS Interface. Although required to run VCS, this `initial.attributes` file contains predefined settings to aid the beginning user of VCS. The the file must be:

```
/$HOME/PCMDI_GRAPHICS/initial.attributes
```

The contents of the `initial.attributes` file can be customized by the user.

Example of Use:

```
a=vcs.init()
...

a.saveinitialfile()
```

scatter(self, *args, **parms)

Function: `scatter` # Generate a scatter plot

Description of Function:

Generate a scatter plot given the data, scatter graphics method, and template. If no scatter class object is given, then the `default` graphics method is used. Similarly, if no template class object is given, then the `default` template is used.

Example of Use:

```
a=vcs.init()
a.show('scatter') # Show all the existing
sct=a.getscatter('quick') # Create instance of 'quick'
a.scatter(array, sct) # Plot array using specified
# template
a.clear() # Clear VCS canvas
a.scatter(array, sct, template) # Plot array using specified
```

scriptobject(self, obj, script_filename=None, mode=None)

Function: `scriptobject` # Script a single primary or secondary

Description of Function:

Save individual attributes sets (i.e., individual primary objects and/or secondary class objects). These attribute sets are saved in the user's current directory.

Note: If the filename has a `".py"` at the end, it will

Python script. If the filename has a ".scr" at the end, it will produce a VCS script. If neither extensions are given, it will default a Python script will be produced.

Note: Mode is either "w" for replace or "a" for append.

Note: VCS does not allow the modification of 'default' attributes; it will not allow them to be saved as individual script files. However, a 'default' attribute set that has been copied to a different name can be saved as a script file.

Example of Use:

```
a=vcs.init()
l=a.getline('red')          # To Modify an existing line object
i=x.createisoline('dean')  # Create an instance of default
...
x.scriptsingle(l,'line.scr','w') # Save line object as a VCS script
x.scriptsingle(i,'isoline.py')  # Save isoline object as a Python script
```

scriptrun(self, *args)

```
#####
#
# Import old VCS file script commands into CDAT.
#
#####
```

scriptstate(self, script_name)

Function: scriptstate # Save state of VCS

Description of Function:

The VCS scripting capability serves many purposes. It allows saving the system state for replay in a later session; to save primary element attributes for use in later visual presentations; to save a record of interactive operations for replay; or to recover from a

Example of Use:

```
a=vcs.init()
...
a.scriptstate(script_filename)
```

set(self, *args)

Function: set

Description of Function:

Set the default VCS primary class objects: template and graphic. Keep in mind the template, determines the appearance of each element; the graphic method specifies the display technique; and the data is to be displayed. Note, the data cannot be set with this

Example of Use:

```
a=vcs.init()
```

```
a.set('isofill','quick') # Changes the default graphics me
a.plot(array)
```

setcolorcell(self, *args)

Function: setcolorcell

Description of Function:

Set a individual color cell in the active colormap. If def
the active colormap, then return an error string.

If the the visul display is 16-bit, 24-bit, or 32-bit True
of the VCS Canvas is made evertime the color cell is chang

Note, the user can only change color cells 0 through 239 a
value must range from 0 to 100. Where 0 represents no col
and 100 is the greatest color intensity.

Example of Use:

```
a=vcs.init()
a.plot(array,'default','isofill','quick')
a.setcolormap("AMIP")
a.setcolorcell(11,0,0,0)
a.setcolorcell(21,100,0,0)
a.setcolorcell(31,0,100,0)
a.setcolorcell(41,0,0,100)
a.setcolorcell(51,100,100,100)
a.setcolorcell(61,70,70,70)
```

setcolormap(self, *args)

Function: setcolormap

Description of Function:

It is necessary to change the colormap. This routine will
color map.

If the the visul display is 16-bit, 24-bit, or 32-bit True
of the VCS Canvas is made evertime the colormap is changed

Example of Use:

```
a=vcs.init()
a.plot(array,'default','isofill','quick')
a.setcolormap("AMIP")
```

setcontinentstype(self, *args)

Function: setcontinentstype

Description of Function:

One has the option of using continental maps that are pred
are user-defined. Predefined continental maps are either i
or are specified by external files. User-defined continent
specified by additional external files that must be read a

The continents-type values are integers ranging from 0 to 5
0 signifies "No Continents"
1 signifies "Fine Continents"
2 signifies "Coarse Continents"
3 signifies "United States" (with "Fine Continents")
4 signifies "Political Borders" (with "Fine Continents")
5 signifies "Rivers" (with "Fine Continents")

Values 6 through 11 signify the line type defined by the following
data_continent_other7 through data_continent_other12.

Example of Use:

```
a=vcs.init()
a.setcontinentstype(3)
a.plot(array, 'default', 'isofill', 'quick')
```

show(self, *args)

Function: show

Description of Function:

Show the list of VCS primary and secondary class objects.

Example of Use:

```
a=vcs.init()
a.show('boxfill')
a.show('isofill')
a.show('line')
a.show('marker')
a.show('text')
```

showbg(self, *args)

Function: showbg

Description of Function:

This function displays graphics segments, which are currently
on the VCS Canvas. That is, if the plot function was called in
background mode), then the plot is produced in the frame buffer
user. In order to view the graphics segments, this function transfers
the frame buffer to the VCS Canvas, where the graphics can be viewed.

Example of Use:

```
a=vcs.init()
a.plot(array, bg=1)
x.showbg()
```

syncP(self, *args)

taylordiagram(self, *args, **parms)

Function: taylordiagram

Generate an

Description of Function:

Generate a taylordiagram plot given the data, taylordiagram

template. If no `taylordiagram` class object is given, then `graphics` method is used. Similarly, if no `template` class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('taylordiagram')           # Show all the existing t
td=a.gettaylordiagram()          # Create instance of 't
a.taylordiagram(array,td)        # Plot array using spec
                                #   template
a.clear()                         # Clear VCS Canvas
a.taylordiagram(array,td,template) # Plot array using spec
```

```
templateeditor(self, template_name='default', template_orig_name='default', plot=None, gui_parent=None)
#####
#
# Template Editor Graphical User Interface wrapper for VCS.
#
#####
```

text = `textcombined`(self, *args, **parms)

textcombined(self, *args, **parms)

Function: `text` or `textcombined` # Generate a `textcombined` object

Description of Function:

Plot a `textcombined` segment on the `Vcs Canvas`. If no `textcombined` object is given, then an error will be returned.

Example of Use:

```
a=vcs.init()
a.show('texttable')              # Show all the existing t
a.show('textorientation')        # Show all the existing t
tt=a.gettext('std','7left')      # Create instance of 'std
tt.string = 'Text1'              # Show the string "Text1"
tt.font=2                        # Set the text size
tt.color = 242                   # Set the text color
tt.angle = 45                    # Set the text angle
tt.x=[[0.0,2.0,2.0,0.0,0.0], [0.5,1.5]] # Set the x value points
tt.y=[[0.0,0.0,2.0,2.0,0.0], [1.0,1.0]] # Set the y value points
a.text(tt)                       # Plot using specified text
```

Optionally you can pass a string, the coordinates and any key words.

Example:

```
x.plot('Hi', .5, .5, color=241, angle=45)
```

update(self, *args)

Function: `update` # Update the `VCS Canvas`.

Description of Function:

If a series of commands are given to `VCS` and the `Canvas Mode` is set to `manual`, then use this function to update the `plot(s)`.

manually.

Example of Use:

...

```
a=vcs.init()
a.plot(s, 'default', 'boxfill', 'quick')
a.mode = 0 # Go to manual mode
box=x.getboxfill('quick')
box.color_1=100
box.xticlabels('lon30', 'lon30')
box.xticlabels('', '')
box.datawc(1e20, 1e20, 1e20, 1e20)
box.datawc(-45.0, 45.0, -90.0, 90.0)

a.update() # Update the change
```

updateVCSsegments(self, *args)

```
#####
#
# Touch all segments displayed on the VCS Canvas.
#
#####
```

updateorientation(self, *args)

Example of Use:

```
a=vcs.init()
x.updateorientation()
```

vector(self, *args, **parms)

Function: vector # Generate a vector plot

Description of Function:

Generate a vector plot given the data, vector graphics method, and template. If no vector class object is given, then the 'default' graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('vector') # Show all the existing
vec=a.getvector('quick') # Create instance of 'vector'
a.vector(array, vec) # Plot array using specified
# template
a.clear() # Clear VCS canvas
a.vector(array, vec, template) # Plot array using specified
```

xvsy(self, *args, **parms)

Function: xvsy # Generate a XvsY plot

Description of Function:

Generate a XvsY plot given the data, XvsY graphics method, and template.

template. If no XvsY class object is given, then the 'default' graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('xvsv')           # Show all the existing X
xy=a.getxvsv('quick')    # Create instance of 'qu
a.xvsv(array,xy)         # Plot array using specif
#           template
a.clear()                # Clear VCS canvas
a.xvsv(array,xy,template) # Plot array using specif
```

xvsv(self, *args, **parms)

Function: xvsv # Generate a Xyvsy plot

Description of Function:

Generate a Xyvsy plot given the data, Xyvsy graphics method and template. If no Xyvsy class object is given, then the 'default' graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('xyvsv')          # Show all the existing
xyy=a.getxyvsv('quick')  # Create instance of 'qu
a.xyvsv(array,xyy)       # Plot array using speci
#           template
a.clear()                # Clear VCS canvas
a.xyvsv(array,xyy,template) # Plot array using speci
```

yvsv(self, *args, **parms)

Function: yvsv # Generate a Yxvsx plot

Description of Function:

Generate a Yxvsx plot given the data, Yxvsx graphics method and template. If no Yxvsx class object is given, then the 'default' graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.

Example of Use:

```
a=vcs.init()
a.show('yxvsv')          # Show all the existing
yxx=a.getyxvsv('quick')  # Create instance of 'qu
a.yxvsv(array,yxx)       # Plot array using speci
#           template
a.clear()                # Clear VCS canvas
a.yxvsv(array,yxx,template) # Plot array using speci
```

class *animate_obj*

Function: animate

Description of Function:

Animate the contents of the VCS Canvas. The animation can also be done from the animation GUI. (See VCDAT for more details.)

See the animation GUI documentation located at URL:
<http://www-pcmdi.llnl.gov/software/vcs>

Example of Use:

```
a=vcs.init()  
a.plot(array, 'default', 'isofill', 'quick')  
a.animate()
```

Methods defined here:

```
__init__(self, vcs_self)  
#####  
# Initialize the animation flags  
#####
```

```
close(self)  
#####  
# Close the animate session  
#####
```

```
create(self, parent=None, min=None, max=None, save_file=None, thread_it=1)  
#####  
# Create the animation images. If min or max is None, then  
# the animator will find the min and max values from the data  
# If min and max are set to 1e20, then no min and max animati  
# value is used (i.e., each animation frame will have differe  
# min and max values. If min and max are set by the user, the  
# these values are used for the animation min and max.  
#  
# If you are running animation from a program, set thread_it  
# This will cause the Python program to wait for the create f  
# to finish before moving onto the next command line.  
#####
```

```
creating_animation_flg(self)  
#####  
# Creating animation flag  
#####
```

```
direction(self, value=1)  
#####  
# Set the direction of the animation:  
# Value 1 -> forward, 2 -> backward  
#####
```

```

frame(self, value=1)
#####
# View the specified animation frame
#####

gui(self, gui_parent=None, transient=0)
#####
# Pop up the animation GUI
#####

horizontal(self, value=0)
#####
# Pan the zoomed animation or frame in the x (or horizontal)
# Value ranges from -100 to 100
#####

load_from_file(self, parent=None, load_file=None, thread_it=1)
#####
# Load animation from a stored Raster file.
#####

mode(self, value=1)
#####
# Mode sets the cycle, forth and back, or animate once
# Value: 1 -> cycle, 2 -> animate once, and 3 -> forth and ba
#####

number_of_frames(self)
#####
# Return the number of animate frames
#####

pause(self, value=1)
#####
# Pause the animation loop
# Value ranges from 0 to 100
#####

restore_min_max(self)
#####
# Restore min and max values
#####

run(self)
#####
# Run or start the animation
#####

run_animation_flg(self)
#####
# Run animation flag
#####

```

```

save_original_min_max(self)
#####
# Save original min and max values
#####

set_animation_min_max(self, min, max, i)
#####
# Set the animation min and max values
#####

stop(self)
#####
# Stop the animation
#####

update_animate_display_list(self)
#####
# Update the animation display list
#####

vertical(self, value=0)
#####
# Pan the zoomed animation or frame in the y (or vertical) di
# Value ranges from -100 to 100
#####

zoom(self, value=1)
#####
# Zoom in on the animation
# Value ranges from 0 to 20
#####

```

Functions

```

change_date_time(tv, number)
#####
#
# Primarily used for resetting the animation date and time string.
#
#####

```

dictionarytovclist(dictionary, name)

finish_queued_X_server_requests(self)

Wait for the X server to execute all pending events.

If working with C routines, then use BLOCK_X_SERVER found in the VCS module routine to stop the X server from continuing. Thus, eliminating the asynchronous errors.

Data

StringTypes = (<type 'str'>, <type 'unicode'>)
called_initial_attributes_flg = 0